New Standards from W3C: XPath, XQuery, and XSLT

Don Chamberlin
IBM Almaden Research Center
Feb. 15, 2007
New from W3C

- Jan. 23, 2007: three new W3C "Recommendations"
  - **XQuery 1.0**: a new XML query language
  - **XSLT 2.0**: a major enhancement to an existing standard for stylesheets and transforms
  - **XPath 2.0**: the common subset of XQuery and XSLT

- **XQuery and XSLT share:**
  - data model
  - function library
  - type system
  - navigation syntax

- These new standards are motivated by convergence of two types of information: documents and data
Evolution of document markup

- "Blue-pencil" instructions to typesetter
- Appearance-related commands in word processors
- Separation of content from appearance
  - Content marked up with generic tags (SGML)
  - Appearance controlled by "style sheets" (DSSSL)
- The explosion of hypertext and the Web (early '90's)
  - HTML: a specific vocabulary of tags for hypertext
  - XML: simpler than SGML, more flexible than HTML
Evolution of databases

- Early databases relied on explicit "navigation"
- Since about 1980, most business data has been relational
  - Data in tables, uniform rows and columns
  - Data has no intrinsic order
  - Automatic optimization of access paths
- A standard language: SQL

```sql
SELECT price * qty
FROM parts
WHERE name = "Bolt"
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>PRICE</th>
<th>QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bolt</td>
<td>0.75</td>
<td>300</td>
</tr>
<tr>
<td>Nut</td>
<td>0.12</td>
<td>300</td>
</tr>
</tbody>
</table>
Convergence of data and documents

- The Web led to new requirements for business data
  - purchase orders, medical records, insurance records, etc.

- Much of this data looks like "documents"
  - Intrinsic order
  - Heterogeneous (every instance is different)
  - Sparse
  - Hierarchic

- Databases need a self-describing data format
  - XML is the obvious choice
  - Metadata mixed with data as "tags"
  - All major database vendors are investing in XML
Beginnings of XML Query

- QL '98 Conference, Boston (W3C)
  - Resulted in ~50 proposals for an XML query language

- The XML Query Working Group
  - Chartered by W3C in October 1999
  - Representatives from about 30 companies
  - Studied QL '98 proposals and generated some new ones
  - Also looked at possible extensions to SQL
  - Decided to develop a new language: "XQuery"
Principles of XQuery Design

- **Closure**
  - Define a data model and a set of operators that are closed under the data model

- **Compositionality**
  - XQuery consists of several kinds of expressions
  - Every expression can be evaluated without side effects
  - Expressions can be composed with full generality

- **Compatibility with existing XML Standards**
  - Type system of XML Schema
  - Naming conventions of XML Namespaces
  - Navigation syntax of XPath (shared with XSLT)
The XQuery Data Model (XDM)

XML Document
(Linear text)

Parsing
Optional validation to add data types

XDM Instance
(Nodes and Atomic Values)

Serialization

XQuery
<?xml version = "1.0"?>
<!-- Requires one trained person -->
<procedure title = "Removing a light bulb">
  <time unit = "sec">15</time>
  <step>Grip bulb.</step>
  <step>
    Rotate it
    <warning>slowly</warning>
    counterclockwise.
  </step>
</procedure>
... and its XDM Representation

---

**A**

**A**

**step**

**E**

**E**

**step**

**E**

**step**

**E**

**warning**

**E**

**counterclockwise.**

**E**

**Grip bulb.**

**E**

**Rotate it**

**E**

**slowly**

**E**

**title="Removing a light bulb"**

**D**

**procedure**

**E**

**time**

**E**

**unit="sec"**

**E**

**15**
Learning from existing standards

- **XSLT (XML Stylesheet Language: Transforms)**

- **Used for transforming XML documents**
  - Often into HTML for display or printing
  - Sometimes into another type of XML document
  - Sometimes into something else (PDF etc.)

- **XSLT:**
  - uses an XML syntax
  - is based on matching "patterns"
    (each pattern can generate some output)
  - uses XPath for navigation (finding patterns)
XPath

- XPath is used for finding nodes that match a pattern
- XPath can find things but cannot create new things
- The simplest form of XPath looks like a downward path with optional predicates
- Each step returns a list of nodes in document order
- These nodes in turn provide context for the next step
- Example:
  
  /company[@location = "Denver"]
  /employee[secretary]/language[1]
XPath has 3 kinds of predicates

- Boolean expressions:
  `book[author = "Mark Twain"]`

- Numeric expressions:
  `chapter[2]`

- Existence tests:
  `book[appendix]`
  `person[married]` (Tests existence, not value!)

- It's not always possible to distinguish these statically
  - Makes optimization difficult
XPath design philosophy

- Few types
  - Boolean, String, Number, Node Set

- Few errors (do something reasonable and keep moving)
  - Cast to the needed type (very permissive)
  - Use the first element in a list if you need only one

- Implicit operations
  - Extract the value of a node when you need it
  - Comparisons based on existential quantifiers
    
    ```
    bonus > salary means:
    some b in bonus, s in salary
    satisfies number(b) > number(s)
    ```
Decision to use XPath in XQuery

- Adopt XPath as a navigation syntax
- Update XPath to the type system of XML Schema
- Use XPath semantics for arithmetic, comparisons, etc.
- Invent other composable expressions for additional functionality (constructors, etc.)
- A path is a "leaf" of the XQuery expression tree
Some implications of using XPath

- Case-sensitive language

- No reserved words
  - `return` is a name

- Can't use `/` for division
  - `a/b` vs. `a div b`

- `a-b` is a name
  - `a-b` vs. `a - b`
Some XQuery expressions

- **Iteration**: for $x$ in expr1 return expr2
- **Conditional**: if (test) then expr1 else expr2
- **Existential**: some $x$ in expr1 satisfies test2
- **Universal**: every $x$ in expr1 satisfies test2
- **Set operations**: union, intersect, except
- **Constructors**:
  
  <greeting>Hello</greeting>

  <revenue>{$price * $quantity}</revenue>
Meanwhile, back at XSLT

- Updating XPath to the type system of XML Schema
- Extending XPath with new kinds of expressions (if-then-else, set operations, existential and universal quantifiers, iterating functions over sequences, etc.)
- Agreement (2001)
  - 2 working groups get "joint custody" of XPath-2
  - Common functionality to be pushed into XPath-2
  - Path expression is no longer a "leaf" (full compositionality)
  - Working groups agree to meet jointly with each other and with Schema
Fun with XPath 1.0

- a[b = 5] returns a-elements that have any b-child with value 5
- a[b+0 = 5] returns a-elements whose first b-child has value 5
- a[b-0 = 5] returns a-elements that have any child named "b-0" with value 5
Fun with XPath 1.0, continued

- `//person[8]`
  returns the eighth person in document order

- `//person[shoesize]`
  returns all persons who have at least one shoesize

- `//person[shoesize + 0]`
  returns persons whose position in the list of persons is equal to their (first) shoesize
Fun with XPath 1.0, continued

- Comparisons:
  - "4" = "4.0" returns False (compared as strings)
  - "4" >= "4.0" returns True  (compared as numbers)
  - "4" <= "4.0" returns True  (compared as numbers)

- These elements are "equal" according to the "=" operator:

  <book>
    <author> Mark Twain </author>
    <title> Huckleberry Finn </title>
  </book>

  <book>
    <title> Mark Twain </title>
    <author> Huckleberry Finn </author>
  </book>
What did we do about this?

- XQuery wanted strong and consistent typing
  - Adding a number to a list is an error
  - Comparing a number to a string is an error
  - Strings are always compared as strings, not numbers
  - Deep-equal function defined for comparing elements

- XSLT wanted backward compatibility

- Both languages wanted to be supersets of XPath-2

- The compromise:
  - XPath-2 has a "compatibility mode"
  - XQuery always turns it off (not compatible with XPath-1)
  - XSLT gives the user a choice
Issue: transitive comparisons

- XPath comparison ops:  =  !=  <  <=  >  >=

- Existential semantics
  - author = "Gray" is true if *any* author is Gray
  - Not good for exact comparisons

- Not transitive
  - (1, 2) = (2, 3) and (2, 3) = (3, 4) but (1, 2) != (3, 4)
  - (1, 4) > (2, 3) and (1, 4) < (2, 3) are both true
  - Not good for ordering, grouping

- We added "value comparisons": eq ne lt le gt ge
  - Transitive
  - Raise an error if either operand has multiple values
Issue: errors and indeterminacy

- An expression may evaluate its operands in any order

- Some expressions may either return a result or raise an error
  
  - `bonus > 5 and salary div 0 > 6`
  
  - `some $c in $cars satisfies $c/price div $c/mileage < 1000`

  - `product[price > 100]` (allowed to use an index)

- General principle: No need to search for data that could only raise errors
**Issue: types**

- Where do types come from?
  - Named typing vs. structural typing
  - What is this? `<a>12</a>`
  - Each operator has its own rules for untyped data

- What is the syntax of a type?
  - Used in function signatures, node tests, cast expressions
  - Simplified:
What did we do right?

- We took existing standards seriously (XPath, Schema, Namespaces)
- XQuery operates on XML in its own data model
  - No need to transform XML into something else
  - Much less code than conventional XML apps
  - Rapid prototyping, apps are easy to build and change
- Declarative, functional language (optimizable)
- Gracefully integrates navigation with construction
- Usable in many environments
  - Typed and untyped data
  - Stand-alone or with a host language
  - With file systems, databases, streams and feeds
What did we do wrong?

- We took existing standards seriously
  - Inherited all the complexity and foibles of XPath + Schema + Namespaces
  - $x[y]$ might be a positional predicate, or might not
  - Schema has 44 built-in types, two kinds of inheritance, "substitution groups", "nillable" elements, etc.

- Our syntax is fragile and sometimes ugly
  - No reserved words: return is a query
  - What is this? delete union + 2
  - Double-token approach: do delete

- We left out some important things
  - Updates, grouping, error handling, text search

- We took way too long
Why did it take so long?

- We took existing standards seriously
  - We had to reconcile XPath with XML Schema
  - We spent a lot of time on the type system

- We published several working drafts per year and responded to public comments
  - ~2000 public comments during "last call" period

- We developed a shared function library (128 functions)

- We built a comprehensive test suite
  - More than 15,000 test cases
  - 14 implementations have submitted test results
  - 11 have demonstrated at least 98% conformance
Where are we now?

- XPath-2 adapts XPath to the Schema type system and adds many new operators:
  for if-then-else some every intersect eq etc.

- XQuery includes all of XPath-2 plus: constructors, FLWOR, user-defined functions, etc.

- XSLT-2 includes all of XPath-2 plus: grouping, user-defined functions, validation, etc.

- Both languages share a new function library
Comparison of XQuery and XSLT

- XQuery and XSLT are (roughly) equivalent in power
  - Both are Turing-complete languages
  - Open-source translator available from XQuery into XSLT

- Some things are easier to do in one than the other
  - XSLT is more oriented toward documents, formatting, whole-document transformations
  - XQuery more oriented toward data, extraction of small query results, SQL users
XQuery and XSLT

- XSLT is older and more established
  - XSLT: 20M Google hits, 25 books on Amazon
  - XQuery: 5M Google hits, 8 books on Amazon

- XQuery is gaining traction
  - W3C lists 48 XQuery implementations (some partial)
  - Some are databases, XML or hybrid
  - Some are data integrators (merge and transform XML data)
  - Several are free and open-source
  - FLWOR Foundation: www.flworfound.org
The Query working group has been rechartered

Working drafts nearing last call:
- XQuery Update Facility (insert, delete, replace, rename)
- Full-text search (ranking, stemming, synonyms, etc.)

Now in the requirements stage:
- XQuery 1.1 (grouping, try/catch, etc.)
- Scripting extensions (sequential execution, assignments, while-loops, local variables, etc.)